

## Contents

### Section 1: Syntax Extensions

- 1.1 Program heading
- 1.2 Declaration ordering
- 1.3 Comment brackets
- 1.4 ELSE in CASE statement
- 1.5 EXIT statement
- 1.6 EXTERNAL procedures and functions
- 1.7 FORTRAN procedures and functions

### Section 2: Low-Level Interface

- 2.1 Octal (Base 8) Numbers
- 2.2 Unsigned Integers
- 2.3 AND, OR, NOT operators on Integer
- 2.4 Absolute memory addressing (ORIGIN)
- 2.5 Address operator (@)
- 2.6 Embedded assembly code

### Section 3: I/O Support Extensions

- 3.1 Reset()/Rewrite() standard procedures
- 3.2 Seek() procedure
- 3.3 Break() procedure
- 3.4 Close() procedure
- 3.5 Readln() Array of Char
- 3.6 Write() Array of Char
- 3.7 Write() Octal (Base 8)
- 3.8 Interactive I/O

### Section 4: Additional Predefined Functions

- 4.1 Time
- 4.2 Exp10() and Log()

### Section 5: Non-Standard Language Elements

- 5.1 Pack()/Unpack() not available
- 5.2 Program parameters
- 5.3 Identifier scope rules
- 5.4 Read()/Write() Text files only
- 5.5 Eof() not accurate (RT11, RSTS only)

### Section 6: Implementation Definitions

- 6.1 Identifiers
- 6.2 Standard type Integer
- 6.3 Standard type Real
- 6.4 Standard type Char
- 6.5 Standard type Text
- 6.6 SET types
- 6.7 New() and Dispose()
- 6.8 Procedural Parameters
- 6.9 Implementation Limitations
- 6.10 Error Detection

TABLE A: Predefined Identifiers

TABLE B: Reserved Words

1. The first part of the report is a general introduction to the subject of the study. It discusses the importance of the study and the objectives of the research.

2. The second part of the report is a detailed description of the methodology used in the study. It includes information about the sample size, the data collection methods, and the statistical analysis techniques.

3. The third part of the report is a presentation of the results of the study. It includes tables and graphs showing the data and the findings of the research.

4. The fourth part of the report is a discussion of the results and their implications. It discusses the strengths and limitations of the study and provides suggestions for future research.

5. The fifth part of the report is a conclusion and a summary of the findings. It provides a final statement on the results of the study and the overall conclusions.

The first part of the report is a general introduction to the subject of the study. It discusses the importance of the study and the objectives of the research.

The second part of the report is a detailed description of the methodology used in the study. It includes information about the sample size, the data collection methods, and the statistical analysis techniques.

The third part of the report is a presentation of the results of the study. It includes tables and graphs showing the data and the findings of the research.

The fourth part of the report is a discussion of the results and their implications. It discusses the strengths and limitations of the study and provides suggestions for future research.

The fifth part of the report is a conclusion and a summary of the findings. It provides a final statement on the results of the study and the overall conclusions.

The first part of the report is a general introduction to the subject of the study. It discusses the importance of the study and the objectives of the research.

The second part of the report is a detailed description of the methodology used in the study. It includes information about the sample size, the data collection methods, and the statistical analysis techniques.

The third part of the report is a presentation of the results of the study. It includes tables and graphs showing the data and the findings of the research.

The fourth part of the report is a discussion of the results and their implications. It discusses the strengths and limitations of the study and provides suggestions for future research.

The fifth part of the report is a conclusion and a summary of the findings. It provides a final statement on the results of the study and the overall conclusions.

The first part of the report is a general introduction to the subject of the study. It discusses the importance of the study and the objectives of the research.

The second part of the report is a detailed description of the methodology used in the study. It includes information about the sample size, the data collection methods, and the statistical analysis techniques.

The third part of the report is a presentation of the results of the study. It includes tables and graphs showing the data and the findings of the research.

The fourth part of the report is a discussion of the results and their implications. It discusses the strengths and limitations of the study and provides suggestions for future research.

The fifth part of the report is a conclusion and a summary of the findings. It provides a final statement on the results of the study and the overall conclusions.

The first part of the report is a general introduction to the subject of the study. It discusses the importance of the study and the objectives of the research.



## 1.0 Syntax Extensions

This section describes extensions to the formal structure of Pascal which are of general utility.

### 1.1 Program heading

The program heading is optional in OMSI Pascal-1 programs, and it may be omitted entirely. If the program heading appears, the program name will be printed on each page of the program listing. The first six characters of the name will be used as the external name of the object module. Parameters appearing in the program heading are ignored.

### 1.2 Declaration ordering

The ordering of global declaration sections (CONST, TYPE, VAR, LABEL) is extended in OMSI Pascal-1. Declaration sections may appear more than once and in any order, so long as identifiers are defined before being used.

One application of this is the concatenation of source modules with main programs which provides a primitive source library capability.

Example - compiler input PLOT,MAIN:

```
(* define source module PLOT *)
VAR ...    (* global plotter variables *)
PROCEDURE (* and plotter functions *)
PROCEDURE ...
(* end of plotter module *)

(* program file MAIN *)
VAR ...    (* global variables *)
BEGIN (* main program code *) END.
```

### 1.3 Comment brackets

OMSI Pascal-1 provides three forms of comment brackets: the Standard braces {...}, the Standard alternate for upper-case terminals (\*...\*), and the additional form /\*...\*/. These may be interchanged freely - it is not necessary for opening and closing comment brackets to have the same form. Comments may not be nested.

ORIGINAL ARTICLES

1. The Effect of the Diet on the Blood Sugar in the Normal Individual  
2. The Effect of the Diet on the Blood Sugar in the Diabetic Individual

3. The Effect of the Diet on the Blood Sugar in the Diabetic Individual  
4. The Effect of the Diet on the Blood Sugar in the Diabetic Individual

5. The Effect of the Diet on the Blood Sugar in the Diabetic Individual  
6. The Effect of the Diet on the Blood Sugar in the Diabetic Individual

7. The Effect of the Diet on the Blood Sugar in the Diabetic Individual  
8. The Effect of the Diet on the Blood Sugar in the Diabetic Individual

ORIGINAL ARTICLES

9. The Effect of the Diet on the Blood Sugar in the Diabetic Individual  
10. The Effect of the Diet on the Blood Sugar in the Diabetic Individual

11. The Effect of the Diet on the Blood Sugar in the Diabetic Individual  
12. The Effect of the Diet on the Blood Sugar in the Diabetic Individual

ORIGINAL ARTICLES

13. The Effect of the Diet on the Blood Sugar in the Diabetic Individual  
14. The Effect of the Diet on the Blood Sugar in the Diabetic Individual



Examples:

```
(* This is a valid comment */  
{ This is (* not *) a valid comment }
```

1.4 ELSE clause in CASE statements

DMSI Pascal-1 allows an optional ELSE clause to appear in a CASE statement. It indicates a statement which is to be executed if the CASE selector expression does not match the value of any CASE label. If included, the ELSE clause follows all other statements inside the CASE statement. If no ELSE clause appears and no statement is selected, control passes to the statement following the CASE statement.

Example:

```
repeat  
  Readln(Ch);  
  case Ch of  
    'A','a': Append;  
    'D','d': Delete;  
    'I','i': Insert;  
    'N','n': Newfile;  
    'Q','q': ;  
    else Writeln('"' ,Ch,'" is not a legal command');  
  end;  
until (Ch = 'Q') or (Ch = 'q');
```

1.5 EXIT statement

The EXIT statement terminates the immediately enclosing iterative statement (WHILE, REPEAT, FOR).

The EXIT statement is included for compatibility with previous versions of DMSI Pascal-1. Its use is not recommended in programs intended to be portable.

Example (table search):

```
Found := False;  
for I := 1 to Tablesize do  
  if Table[I]=Key  
    then begin  
      Found := True;  
      exit;  
    end;  
end;
```

Page 1

1. The purpose of this document is to provide information regarding the activities of the [redacted] in the [redacted] area.

### 2. Background Information

The [redacted] has been active in the [redacted] area since [redacted]. It has been observed that the [redacted] has been involved in a variety of activities, including [redacted]. The [redacted] has been observed to be involved in the [redacted] of [redacted] and [redacted]. The [redacted] has been observed to be involved in the [redacted] of [redacted] and [redacted]. The [redacted] has been observed to be involved in the [redacted] of [redacted] and [redacted].

3. Findings

- 1. The [redacted] has been observed to be involved in the [redacted] of [redacted] and [redacted].
- 2. The [redacted] has been observed to be involved in the [redacted] of [redacted] and [redacted].
- 3. The [redacted] has been observed to be involved in the [redacted] of [redacted] and [redacted].
- 4. The [redacted] has been observed to be involved in the [redacted] of [redacted] and [redacted].
- 5. The [redacted] has been observed to be involved in the [redacted] of [redacted] and [redacted].

4. Conclusion

The [redacted] has been observed to be involved in the [redacted] of [redacted] and [redacted].

### 5. Recommendations

It is recommended that the [redacted] be monitored closely for any further activities. It is also recommended that the [redacted] be kept informed of any developments in the [redacted] area. It is further recommended that the [redacted] be kept informed of any developments in the [redacted] area. It is further recommended that the [redacted] be kept informed of any developments in the [redacted] area.

6. References

- 1. [redacted]
- 2. [redacted]
- 3. [redacted]
- 4. [redacted]
- 5. [redacted]



### 1.6 EXTERNAL Procedures and Functions

The keyword **EXTERNAL** provides access to separately compiled subroutines and to program libraries and overlay facilities. **EXTERNAL** appears in the place of a procedure or function body to indicate that the procedure or function is compiled separately.

The compiler will generate references to an external (global) symbol. The first six characters of the procedure or function identifier must form a unique external symbol. References to an external procedure or function are resolved at link or task build time.

Note that the compiler is unable to check parameter types at an external interface.

Examples:

```
procedure Erase; external;  
function Rad50(A,B,C: char): Unsigned; external;
```

### 1.7 FORTTRAN Procedures and Functions

The directive '**FORTTRAN**' is similar to the **EXTERNAL** directive. The compiler will generate a calling sequence corresponding to the Digital PDP-11 standard calling sequence, with register 5 (R5) pointing to an argument list. The **FORTTRAN** directive enables calling of external **MACRO** and **FORTTRAN** subroutines. The **FORTTRAN** calling sequence passes parameters by reference, so the corresponding Pascal parameters must be declared as **VAR** parameters.

The **FORTTRAN** directive generates the proper call sequence for **FORTTRAN** subroutines. It does not provide for initialization of the **FORTTRAN** runtime I/O system.

Example:

```
function Difference(var X,Y: Real): Real; fortran;
```

THE UNIVERSITY OF CHICAGO

The University of Chicago is a private research university in Chicago, Illinois. It was founded in 1837 and is one of the oldest and most prominent universities in the United States. The university is known for its academic excellence and its commitment to research and scholarship.

The University of Chicago is a private research university in Chicago, Illinois. It was founded in 1837 and is one of the oldest and most prominent universities in the United States. The university is known for its academic excellence and its commitment to research and scholarship.

The University of Chicago is a private research university in Chicago, Illinois. It was founded in 1837 and is one of the oldest and most prominent universities in the United States. The university is known for its academic excellence and its commitment to research and scholarship.

Library

The University of Chicago is a private research university in Chicago, Illinois. It was founded in 1837 and is one of the oldest and most prominent universities in the United States. The university is known for its academic excellence and its commitment to research and scholarship.

THE UNIVERSITY OF CHICAGO

The University of Chicago is a private research university in Chicago, Illinois. It was founded in 1837 and is one of the oldest and most prominent universities in the United States. The university is known for its academic excellence and its commitment to research and scholarship.

The University of Chicago is a private research university in Chicago, Illinois. It was founded in 1837 and is one of the oldest and most prominent universities in the United States. The university is known for its academic excellence and its commitment to research and scholarship.

Library

The University of Chicago is a private research university in Chicago, Illinois. It was founded in 1837 and is one of the oldest and most prominent universities in the United States. The university is known for its academic excellence and its commitment to research and scholarship.



## 2.0 Low-Level Interface

The low-level interface section describes those OMSI Pascal-1 extensions which are useful to programmers who need access to machine dependent PDP-11 characteristics.

### 2.1 Octal (Base 8) Numbers

Integer constants may be written in octal notation by appending the capital letter 'B' to the number. This applies only to compile-time constants -- runtime integer conversions via Read() are performed using decimal notation.

Example: `const TabCode = 11B; (* ASCII tab character *)`

### 2.2 Unsigned Integers

The predefined type Integer has the subrange (-32768 .. 32767) and uses the PDP-11 signed arithmetic operations. Unsigned integers may be specified with the subrange 0..65535. The compiler will generate the unsigned comparison operations of the PDP-11 and will not detect multiplication and division overflow of unsigned integers.

Unsigned integer operations apply only to integer calculations. I/O conversions and conversions to and from Real values are always signed integer operations.

Example: `type Unsigned=0..65535;`

### 2.3 Logical operations on Integers

The Boolean operators AND, OR, and NOT are extended to Integer operands. The operators perform the Boolean operations on all 16 bits of their operands. This allows testing or setting of individual bits within a word (for instance, status bits within a device register).

Example: `Byte := Ord(Ch) and 377B;`

### 2.4 References to fixed (absolute) memory

OMSI Pascal-1 allows the keyword ORIGIN to appear in variable declarations, associating a variable identifier with a specific memory address. This provides access to fixed memory addresses.



1. [Illegible]

2. [Illegible]

3. [Illegible]

4. [Illegible]

5. [Illegible]

6. [Illegible]

7. [Illegible]

8. [Illegible]

9. [Illegible]

10. [Illegible]

11. [Illegible]

12. [Illegible]

13. [Illegible]

14. [Illegible]



such as device control registers or operating system parameter blocks.

Example (read directly from the RT11 console):

```
const Ready=200B;
var   KbCsr origin 177560B, KbBuff origin 177562B: Integer;
      Ch: Char;
begin
    while (KbCsr and Ready)=0 do (* nothing *)
        Ch := Chr(KbBuff);        (* get character *)
    end;
```

## 2.5 Address operator (@)

DMSI Pascal-1 provides a unary address operator, indicated by the @ character. When applied to a variable of type T, it yields a value of type  $\uparrow T$  (pointer to T). The address operator can be used to link variables into list structures or (more commonly) to pass variable addresses to low-level routines.

Example:

```
var   Buffer: Block; XRLoc origin 446B:  $\uparrow$ Block;
begin
    XRLoc:= @Buffer; (* pass address to RSTS/E *)
end
```

## 2.6 Embedded assembly code

PDP-11 MACRO assembly code may appear at any point in an DMSI Pascal-1 program. Assembly code sections have the form of a Pascal comment, beginning with the %C embedded switch. Any MACRO-11 feature may be used within embedded code. The compiler provides some assistance in accessing Pascal variables, though the programmer is expected to have some understanding of the DMSI Pascal-1 runtime environment. Note that the default radix within a Pascal-produced MACRO file is decimal, not octal.

Example:

```
procedure EmtTrap(N:Integer);
begin
    (*%C
       MOV N(SP),-(SP) ; push parameter N
       EMT 53          ; call EMT handler
    *)
end (*EmtTrap*);
```

1. The purpose of this document is to provide information regarding the security of the system.

2. The information contained herein is classified as CONFIDENTIAL.

3. This document is intended for the use of personnel who are responsible for the security of the system.

4. It is the policy of the organization to protect the security of the system at all times.

5. The following are the security requirements for the system:

- a. All personnel must be properly trained and authorized to access the system.
- b. All data must be protected from unauthorized access and disclosure.
- c. All system components must be maintained in a secure and reliable manner.
- d. All security incidents must be reported and investigated immediately.

6. The security of the system is the responsibility of all personnel who interact with the system.

7. The following are the security procedures for the system:

- a. All personnel must follow the security policies and procedures at all times.
- b. All system components must be tested and validated regularly.
- c. All security incidents must be documented and analyzed thoroughly.
- d. All security updates must be applied promptly.

8. The security of the system is a continuous process that requires ongoing attention and improvement.

9. The information contained herein is classified as CONFIDENTIAL.



### 3.0 I/O Support Extensions

I/O support extensions provide the OMSI Pascal-1 programmer with additional control of the interface to the operating system.

#### 3.1 Reset()/Rewrite() optional parameters

Three additional parameters may appear following the file variable in calls to the Reset() and Rewrite() standard procedures. These optional parameters allow the program to dynamically bind a file variable to an external file and provide status and error information.

The general form is:

```
Reset( F , Filename , DefaultName , Size )
```

where the parameters have these types:

F - any file variable  
Filename - literal string, or (packed) array of Char  
DefaultName - same as Filename  
Size - Integer variable

Reset(F,Filename) connects the file variable F with the external file identified by Filename. Filename conforms to the operating system conventions, and may contain device, filename, extension, and other fields such as PPN/UIC and version number. The Filename parameter may also contain switches specifying access modes or other special characteristics. If the external file does not exist prior to the Reset(), a fatal error will result. Upon successful completion of a Reset(), either the file buffer F↑ will contain the first element of the file, or Eof(F) will be True.

Reset(F,Filename,DefaultName) performs the same function, with DefaultName having the same format as Filename. Fields of the external name which are not specified in Filename are filled from the information in DefaultName. Common default fields are the extension, protection code, and mode switches.

Reset(F,Filename,DefaultName,Size) provides a recovery capability on file open errors. Size must be a variable (VAR parameter). After a successful Reset(), Size contains the length of the file in blocks. If an error occurs, Size is set to negative one (-1).

```
Rewrite( F , Filename , DefaultName , Size )
```

Rewrite() creates a new external file. The optional parameters have the same meaning as in Reset() with one addition: Size specifies the initial storage, in blocks, to be allocated for the file.



THE UNIVERSITY OF CHICAGO PRESS

THE UNIVERSITY OF CHICAGO PRESS  
5401 S. MICHIGAN AVE. CHICAGO, ILL. 60637

THE UNIVERSITY OF CHICAGO PRESS

THE UNIVERSITY OF CHICAGO PRESS  
5401 S. MICHIGAN AVE. CHICAGO, ILL. 60637

THE UNIVERSITY OF CHICAGO PRESS

THE UNIVERSITY OF CHICAGO PRESS

THE UNIVERSITY OF CHICAGO PRESS

THE UNIVERSITY OF CHICAGO PRESS

THE UNIVERSITY OF CHICAGO PRESS  
5401 S. MICHIGAN AVE. CHICAGO, ILL. 60637

THE UNIVERSITY OF CHICAGO PRESS  
5401 S. MICHIGAN AVE. CHICAGO, ILL. 60637

THE UNIVERSITY OF CHICAGO PRESS  
5401 S. MICHIGAN AVE. CHICAGO, ILL. 60637

THE UNIVERSITY OF CHICAGO PRESS  
5401 S. MICHIGAN AVE. CHICAGO, ILL. 60637

THE UNIVERSITY OF CHICAGO PRESS

THE UNIVERSITY OF CHICAGO PRESS  
5401 S. MICHIGAN AVE. CHICAGO, ILL. 60637



Reset() and Rewrite() may be applied to the standard files Input and Output respectively. This will redirect the default input or output streams to the specified file instead of the user terminal. A subsequent Close() will break the connection and reconnect the default file to the terminal.

Example:

```
program Copy; (* copy to printer *)
var Name: array[1..20] of Char;
    Ch: Char; Len: Integer;
begin
    repeat                                     (* Get a Filename and Reset() it *)
        Write('File: ');
        Readln(Name);
        Reset(Input, Name, '.PAS', len)
    until Len < -1;                             (* until not error code *)
    Rewrite(Output, 'LP:'); (* redirect output to printer *)

    while not Eof do begin (* copy Input to Output *)
        while not Eoln do begin
            Read(Ch); Write(Ch);
        end;
        Readln; Writeln;
    end;
end.
```

### 3.2 Seek() procedure

The predefined procedure Seek() causes direct positioning of a file window variable to any desired component of the file.

Seek( F , Index )

F may be of any file type except Text, and must be connected to an external file which supports direct access (typically disk or DECtape). Index is an unsigned integer expression which specifies the desired component. File components are numbered sequentially beginning with one (1). If Index specifies a number greater than the number of components actually present, then Eof(F) is set to True.

To read component N of file F, use:

Seek(F, N); (\* component N is available in F↑ \*)

To write component N, use the sequence:

```
Seek(F, N); (* position to component N *)
F↑ := (); (* assign new value *)
Put(F); (* write component to file *)
```

For the purpose of this study, the following data were collected from the records of the American Medical Association for the year 1934. The data were obtained from the records of the American Medical Association for the year 1934. The data were obtained from the records of the American Medical Association for the year 1934.

THE JOURNAL OF THE AMERICAN MEDICAL ASSOCIATION  
PUBLISHED WEEKLY  
CHICAGO, ILL., U.S.A.

For the purpose of this study, the following data were collected from the records of the American Medical Association for the year 1934. The data were obtained from the records of the American Medical Association for the year 1934. The data were obtained from the records of the American Medical Association for the year 1934.

For the purpose of this study, the following data were collected from the records of the American Medical Association for the year 1934. The data were obtained from the records of the American Medical Association for the year 1934. The data were obtained from the records of the American Medical Association for the year 1934.

For the purpose of this study, the following data were collected from the records of the American Medical Association for the year 1934. The data were obtained from the records of the American Medical Association for the year 1934. The data were obtained from the records of the American Medical Association for the year 1934.



If the Put() in the above sequence is omitted, the effects will be unpredictable and the new data may be lost.

Sequential I/O operations such as Get() and Put() may be mixed with Seek() and will advance the file window to the next component. Reset(F) is equivalent to Seek(F,1).

The direct access extension bypasses the Standard Pascal restriction prohibiting simultaneous read and write access to a file. For this reason, direct access files are identified by the '/Seek' switch which must appear in the Filename or DefaultName field of the associated Reset() or Rewrite().

### 3.3 Break() procedure

For efficiency, OMSI Pascal-1 buffers transmitted data. Break(F) forces the actual transmission of data from a partially filled buffer of file F. This can be useful with interactive terminals, or to guarantee actual transmission of data to a shared disk file.

### 3.4 Close() procedure

Close(F) indicates that the program has completed processing the file F, and that internal buffer storage may be reclaimed. Close(F) removes any connection to an external file, so that Reset(F) or Rewrite(F) must precede any subsequent operations with that file variable.

### 3.5 Readln() Array of Char

Read() and Readln() will read characters from a Text file into a (packed) array of characters. Reading begins at the current file position and continues until either the array is filled, or Eoln() is True, in which case the remainder of the array is filled with blanks.

### 3.6 Write() Array of Char

In accordance with the draft proposed ISO Standard, a Write() procedure call applied to an array of Char will truncate the written string if the field width parameter will not allow the entire string to be written.

THE UNIVERSITY OF CHICAGO PRESS  
5401 S. MICHIGAN AVE. CHICAGO, ILL. 60637

THE UNIVERSITY OF CHICAGO PRESS  
5401 S. MICHIGAN AVE. CHICAGO, ILL. 60637

THE UNIVERSITY OF CHICAGO PRESS  
5401 S. MICHIGAN AVE. CHICAGO, ILL. 60637

THE UNIVERSITY OF CHICAGO PRESS  
5401 S. MICHIGAN AVE. CHICAGO, ILL. 60637

THE UNIVERSITY OF CHICAGO PRESS  
5401 S. MICHIGAN AVE. CHICAGO, ILL. 60637

THE UNIVERSITY OF CHICAGO PRESS  
5401 S. MICHIGAN AVE. CHICAGO, ILL. 60637

THE UNIVERSITY OF CHICAGO PRESS  
5401 S. MICHIGAN AVE. CHICAGO, ILL. 60637

THE UNIVERSITY OF CHICAGO PRESS  
5401 S. MICHIGAN AVE. CHICAGO, ILL. 60637

THE UNIVERSITY OF CHICAGO PRESS  
5401 S. MICHIGAN AVE. CHICAGO, ILL. 60637

THE UNIVERSITY OF CHICAGO PRESS  
5401 S. MICHIGAN AVE. CHICAGO, ILL. 60637

THE UNIVERSITY OF CHICAGO PRESS  
5401 S. MICHIGAN AVE. CHICAGO, ILL. 60637



Example:

```
Write(Buffer:BuffCount); (* write buffered characters *)
```

### 3.7 Write() Octal (Base 8)

Write() will write integers in octal notation if the field width specification is negative.

Example: Write(I:-5); (\* Display octal value of I \*)

### 3.8 Interactive I/O

The Pascal Standard requires that the first element of a file be available as soon as the file is Reset() (the buffer variable F<sup>↑</sup> is assigned a value immediately). This can present serious difficulties when applied to files which are interactive terminals. For example, if the default input file is the user's terminal, the standard can be interpreted to require that the user type the first input character (or line) prior to the execution of the first program statement.

OMSI Pascal-1 takes the following route around the problem. When an interactive file is Reset(), the buffer variable is set to a space and Eoln(F) is set to False, but no actual I/O transmission occurs. Each Read() request then waits for sufficient data to satisfy the request, but no more.

This solves most of the problems with interactive terminals in a predictable manner, but one should note that this approach creates other difficulties. When applied to an interactive file, the following program is unable to distinguish between an empty line and a line containing a single space. This is because Eoln() cannot be set until the end of line character is typed to satisfy the Read() request.

Example: (the standard schema for reading a line of characters)

```
var Line: array[1..72] of Char;  
    Count: Integer;  
begin  
    Count := 0;  
    while not Eoln do begin  
        Count := Count+1;  
        Read(Line[Count]);  
    end;  
    Readln;  
end;
```





#### 4.0 Additional Predefined Functions

OMSI Pascal-1 provides some additional built-in functions.

##### 4.1 Time function

The Time function takes no parameters and returns a real value which corresponds to the current time of day. The Time is represented in hours after midnight, so that 9:30 AM is 9.50 and 1:45 PM is 13.75. The exact resolution of the Time function is dependent on the operating system, but all operating systems provide a resolution of at least one second.

Example:

```
procedure WriteTime;
var Hrs, Mins: Integer;
    AmPm: array[1..2] of Char;
begin
  Mins := Round(Time*60);
  Hrs := Mins div 60;
  Mins := Mins mod 60;
  if (Hrs < 12)
    then AmPm := 'AM'
  else if (Hrs = 12) and (Mins = 0)
    then AmPm := 'M '
  else AmPm := 'PM';
  Write('At the tone the time will be: ');
  Write(((Hrs+11) mod 12 + 1):2);
  Write(':', Mins div 10:1, Mins mod 10:1, AmPm:3);
  Writeln(Chr(7));
end;
```

"At the tone the time will be: 11:56 AM"

##### 4.2 Exp10() and Log() functions

The Exp10() and Log() functions are similar to the standard Exp() and Ln() functions, but with a logarithm base of ten (10).

REPORT OF THE COMMISSIONER OF THE GENERAL LAND OFFICE

FOR THE YEAR ENDING DECEMBER 31, 1907

GENERAL STATEMENT

The General Land Office has the honor to acknowledge the receipt of the report of the Commissioner of the General Land Office for the year ending December 31, 1907, and to express its appreciation of the thorough and complete manner in which the same has been prepared. The report contains a full and complete statement of the work of the office during the year, and a full and complete statement of the financial condition of the office at the close of the year.

FINANCIAL STATEMENT

The following statement shows the financial condition of the General Land Office at the close of the year ending December 31, 1907:

Assets: Cash, \$1,000,000.00; Receivables, \$500,000.00; Land, \$1,500,000.00; Other, \$1,000,000.00. Total, \$4,000,000.00.

Liabilities: Payables, \$1,000,000.00; Other, \$1,000,000.00. Total, \$2,000,000.00.

The following statement shows the financial condition of the General Land Office at the close of the year ending December 31, 1907:

The following statement shows the financial condition of the General Land Office at the close of the year ending December 31, 1907:

GENERAL STATEMENT

The following statement shows the financial condition of the General Land Office at the close of the year ending December 31, 1907:



## 5.0 Non-Standard Language Elements

This section describes the elements of OMSI Pascal-1 which do not conform to the accepted definition of Standard Pascal.

### 5.1 Pack() and Unpack() not available

The reserved word PACKED may appear in type definitions, but it has no meaning in OMSI Pascal-1 programs. Packed types require the same amount of storage as unpacked types. The standard procedures Pack() and Unpack() are not available. The following equivalent FOR statements can be used instead:

```
var A: array[M..N] of T;  
    Z: packed array[P..Q] of T;  
    for J:= P to Q do Z[J]:= A[J-P+I]; { Pack(A,I,Z) }  
    for J:= P to Q do A[J-P+I]:= Z[J]; { Unpack(Z,A,I) }
```

### 5.2 Program Parameters

Program parameters (identifiers appearing in the program heading) have no meaning in OMSI Pascal-1 programs. The program heading may be omitted entirely if desired. External files can be declared by using the Reset() and Rewrite() procedures with optional parameters.

### 5.3 Identifier Scope Rules

In Standard Pascal, the scope of an identifier (that section of the program within which the identifier indicates a particular object) is directly related to the block structure. A definition of an identifier in a procedure, for example, prohibits that identifier from indicating another object throughout the entire procedure.

OMSI Pascal-1 uses a subtly different rule for the scope of an identifier, called 'one-pass' scope, in which a definition of an identifier prohibits only subsequent uses of the identifier within the block from indicating an object outside the block.

The non-standard scope rule is described here for completeness, but it is of little concern to the programmer. Indeed, the majority of Pascal compilers use the identical (incorrect) rule.



THE UNIVERSITY OF CHICAGO  
DIVISION OF THE PHYSICAL SCIENCES

REPORT OF THE COMMITTEE ON THE PHYSICS OF THE ATOM

THE COMMITTEE ON THE PHYSICS OF THE ATOM, which was organized in 1946, has the honor to submit to the Board of the University of Chicago the following report on its activities during the past year.

1. THE PHYSICS OF THE ATOM

The Committee on the Physics of the Atom has been organized to coordinate the activities of the various departments and divisions of the University of Chicago which are engaged in research in the field of atomic physics. The Committee has held several meetings during the past year, and has been successful in securing the cooperation of the various departments and divisions in the study of the problems of atomic physics.

The Committee has also been successful in securing the cooperation of the various departments and divisions in the study of the problems of atomic physics. The Committee has also been successful in securing the cooperation of the various departments and divisions in the study of the problems of atomic physics.

2. THE PHYSICS OF THE ATOM

The Committee on the Physics of the Atom has been organized to coordinate the activities of the various departments and divisions of the University of Chicago which are engaged in research in the field of atomic physics. The Committee has held several meetings during the past year, and has been successful in securing the cooperation of the various departments and divisions in the study of the problems of atomic physics.

3. THE PHYSICS OF THE ATOM

The Committee on the Physics of the Atom has been organized to coordinate the activities of the various departments and divisions of the University of Chicago which are engaged in research in the field of atomic physics. The Committee has held several meetings during the past year, and has been successful in securing the cooperation of the various departments and divisions in the study of the problems of atomic physics.

The Committee on the Physics of the Atom has been organized to coordinate the activities of the various departments and divisions of the University of Chicago which are engaged in research in the field of atomic physics. The Committee has held several meetings during the past year, and has been successful in securing the cooperation of the various departments and divisions in the study of the problems of atomic physics.

The Committee on the Physics of the Atom has been organized to coordinate the activities of the various departments and divisions of the University of Chicago which are engaged in research in the field of atomic physics. The Committee has held several meetings during the past year, and has been successful in securing the cooperation of the various departments and divisions in the study of the problems of atomic physics.



#### 5.4 Read()/Write() Text files only

In the 1978 printing of the Pascal User Manual and Report, the Read() and Write() standard procedures were extended to apply to all file types. This extension has not yet been incorporated into OMSI Pascal-1, so that Read() and Write() are applicable only to files of the standard type Text.

The following substitutions may be used:

For Read(F,V), use: V:=F↑; Get(F);

For Write(F,V), use: F↑:=V; Put(F);

#### 5.5 Eof() not accurate (RT11, RSTS only)

On the RT11 and RSTS operating systems, a file is structured as a sequence of 512 byte blocks. No finer resolution is available as to the end of data in the last block. Therefore, the Eof() standard function can not be relied upon as accurate, and another method (sentinel record, record count) should be used to indicate the end of usable data.

Note that this problem does not apply to Text files, where Eof() is identified correctly.

1. The first part of the document is a letter from the President of the United States to the Congress, dated January 1, 1861. It is a very important document, as it sets out the President's policy for the new year. The letter is written in a very formal and dignified style, and it is one of the most important documents in the history of the United States.

2. The second part of the document is a letter from the President to the Congress, dated January 1, 1861. It is a very important document, as it sets out the President's policy for the new year. The letter is written in a very formal and dignified style, and it is one of the most important documents in the history of the United States.

3. The third part of the document is a letter from the President to the Congress, dated January 1, 1861. It is a very important document, as it sets out the President's policy for the new year. The letter is written in a very formal and dignified style, and it is one of the most important documents in the history of the United States.

4. The fourth part of the document is a letter from the President to the Congress, dated January 1, 1861. It is a very important document, as it sets out the President's policy for the new year. The letter is written in a very formal and dignified style, and it is one of the most important documents in the history of the United States.

5. The fifth part of the document is a letter from the President to the Congress, dated January 1, 1861. It is a very important document, as it sets out the President's policy for the new year. The letter is written in a very formal and dignified style, and it is one of the most important documents in the history of the United States.

6. The sixth part of the document is a letter from the President to the Congress, dated January 1, 1861. It is a very important document, as it sets out the President's policy for the new year. The letter is written in a very formal and dignified style, and it is one of the most important documents in the history of the United States.

7. The seventh part of the document is a letter from the President to the Congress, dated January 1, 1861. It is a very important document, as it sets out the President's policy for the new year. The letter is written in a very formal and dignified style, and it is one of the most important documents in the history of the United States.

8. The eighth part of the document is a letter from the President to the Congress, dated January 1, 1861. It is a very important document, as it sets out the President's policy for the new year. The letter is written in a very formal and dignified style, and it is one of the most important documents in the history of the United States.

9. The ninth part of the document is a letter from the President to the Congress, dated January 1, 1861. It is a very important document, as it sets out the President's policy for the new year. The letter is written in a very formal and dignified style, and it is one of the most important documents in the history of the United States.



## 6.0 Implementation Definitions

This section provides specific details and characteristics of implementation-defined elements of OMSI Pascal-1.

### 6.1 Identifiers

OMSI Pascal-1 permits identifiers to be of any length, and all characters are significant. Lower case letters may be used and are interpreted the same as upper case, so that "name", "Name", and "NAME" are equivalent identifiers.

Due to limitations of the object program file structures, the first six characters of any EXTERNAL or FORTRAN identifier must form a unique external name.

### 6.2 Standard type Integer

The standard type Integer has the range  $(-32768..32767)$ . Unsigned integers may be declared using the subrange notation  $0..65535$ . Note that arithmetic overflow is detected only for multiplication and division of signed integers.

The predefined identifier Maxint has the value 32767.

### 6.3 Standard type Real

Real variables have the standard PDP-11 single or double precision floating point structure, with the range  $1E-38 .. 1E+38$ . Single precision values give 7 decimal digit precision; extended (double precision) values give 15 digit precision. Arithmetic overflow is detected for all real operations, but underflow is ignored and gives a result of zero.

The standard transcendental routines are accurate to 6 decimal digits in single precision, and 15 decimal digits in extended precision.

### 6.4 Standard type Char

OMSI Pascal-1 uses the 7-bit full ASCII character set. Characters are stored as signed bytes with all 8 bits available to the programmer, so that Ord(Char) has the subrange  $(-128..127)$ .



1. Background

The following information was obtained from the files of the [redacted] and is being provided for your information.

2. Summary

The [redacted] has been identified as a [redacted] and is being provided for your information. The [redacted] has been identified as a [redacted] and is being provided for your information.

The [redacted] has been identified as a [redacted] and is being provided for your information. The [redacted] has been identified as a [redacted] and is being provided for your information.

3. Details

The [redacted] has been identified as a [redacted] and is being provided for your information. The [redacted] has been identified as a [redacted] and is being provided for your information.

The [redacted] has been identified as a [redacted] and is being provided for your information.

4. Conclusion

The [redacted] has been identified as a [redacted] and is being provided for your information. The [redacted] has been identified as a [redacted] and is being provided for your information.

The [redacted] has been identified as a [redacted] and is being provided for your information. The [redacted] has been identified as a [redacted] and is being provided for your information.

5. Recommendations

The [redacted] has been identified as a [redacted] and is being provided for your information. The [redacted] has been identified as a [redacted] and is being provided for your information.



### 6.5 Standard type Text

The standard type Text is a file type with components of type Char, with the characters masked to the 7-bit ASCII set, and skipping the null (0) character. On RSX systems, the standard function Eoln() is set by the end of a file record; on RSTS/E and RT11 systems by the LF (10) or ESC (27) character codes.

The standard procedures Read(), Readln(), Write(), Writeln(), and the standard function Eoln() are applicable only to Text files. The Seek() procedure is not recommended for use with Text files.

### 6.6 SET types

OMSI Pascal-1 limits sets to a maximum of 64 elements. The 64 element maximum forms a subrange which is not required to have a lower bound of zero, but may instead be positioned at any 64 element (or smaller) subrange of a base type (for example: 100..150, -25..25).

A set of the standard type Char is equivalent to the set of Chr(32)..Chr(95), which is a subset of ASCII containing the upper case letters, digits, punctuation symbols, and the space character, but lacking the control characters and lower case letters.

### 6.7 New() and Dispose() procedures

In allocating storage for variant records, the New() procedure will allocate memory for the largest variant; any tag field values specified to New() and Dispose() are ignored.

Storage must be explicitly released with Dispose() -- no automatic garbage collection is performed. Storage occupied by variables passed to Dispose() is reclaimed for use by the New() procedure. Dangling pointer references are not detected.

### 6.8 Procedural Parameters

The passing of PROCEDURE and FUNCTION parameters is supported by OMSI Pascal-1 with the syntax described in the Pascal User Manual and Report (the proposed ISO Standard differs in this area).

Predefined procedures and functions are not permitted as procedural parameters. This can be bypassed by declaring a second procedure which calls the standard procedure, and which can itself be used as a procedural parameter.



The first part of the document discusses the importance of maintaining accurate records of all transactions. It emphasizes that proper record-keeping is essential for the integrity of the financial system and for the ability to detect and prevent fraud. The document also notes that records should be kept for a sufficient period of time to allow for a thorough review in the event of an audit or investigation.

The second part of the document discusses the importance of maintaining accurate records of all transactions. It emphasizes that proper record-keeping is essential for the integrity of the financial system and for the ability to detect and prevent fraud. The document also notes that records should be kept for a sufficient period of time to allow for a thorough review in the event of an audit or investigation.

The third part of the document discusses the importance of maintaining accurate records of all transactions. It emphasizes that proper record-keeping is essential for the integrity of the financial system and for the ability to detect and prevent fraud. The document also notes that records should be kept for a sufficient period of time to allow for a thorough review in the event of an audit or investigation.

The fourth part of the document discusses the importance of maintaining accurate records of all transactions. It emphasizes that proper record-keeping is essential for the integrity of the financial system and for the ability to detect and prevent fraud. The document also notes that records should be kept for a sufficient period of time to allow for a thorough review in the event of an audit or investigation.



**Example:**

```
function Sine(X: Real): Real;  
begin  
  Sine:= Sin(X)  
end;
```

**6.9 Implementation Limitations**

The PDP-11 has six general purpose registers. In OMSI Pascal-1, one register (R5) is always allocated for access to global variables, and another (R4) is allocated in some blocks for access to intermediate level variables. The remaining registers are used for integer calculations, address computations, and WITH statement variable access. Each WITH statement uses one register for the duration of the enclosed statement. This implies a maximum nesting of WITH statements of three levels. Complex expression calculations can also exceed the available registers. If the 'Out of registers' error occurs, remove a WITH statement or simplify the indicated expression by calculating intermediate results.

The syntactic nesting of procedures is limited to a depth of 10 levels. There is no implementation restriction on the actual depth of recursion of a program, although unlimited recursion will eventually cause the program to exceed available memory.

**6.10 Error Detection**

OMSI Pascal-1 does not detect the following runtime errors:

- Uninitialized variables
- Subrange bounds exceeded
- Integer overflow
- Real underflow
- Record variant mismatch
- Dereference of NIL pointer

The following runtime errors are detected:

- Stack overflow
- Heap overflow
- Real overflow
- Integer multiply/divide overflow
- Array bounds exceeded
- Dispose() of NIL or duplicate pointer
- Incorrect numeric format
- I/O errors







## Constants

## Types

## Variables

## Functions

### Base 10 Exponential

Time of day

```
Transmit buffered output
Close file
```

Direct access I/O

WATER RESOURCES DIVISION

Report No. 100-100

July 1964

100-100  
100-100  
100-100  
100-100

100-100

100-100

100-100

100-100

100-100

100-100

100-100

100-100

100-100

100-100

100-100

100-100

100-100

100-100

100-100

100-100

100-100

100-100

100-100

100-100

100-100

100-100

100-100

100-100

100-100

100-100

100-100

100-100

100-100

100-100

100-100

100-100

100-100

100-100

100-100

100-100

100-100

100-100

100-100

100-100

100-100

100-100

100-100

100-100

100-100

100-100



Reserved Words  
(\* extensions)

And  
Array  
Begin  
Case  
Const  
Div  
Do  
Downto  
Else  
End  
Exit \*  
External \*  
File  
For  
Fortran \*  
Forward  
Function  
Goto  
If  
In  
Label  
Mod  
Nil  
Not  
Of  
Or  
Origin \*  
Packed  
Procedure  
Program  
Record  
Repeat  
Set  
Then  
To  
Type  
Until  
Var  
While  
With

